(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 043 658 A1

(12) DEMANDE DE BREVET EUROPEEN

(43) Date de publication:
11.10.2000 Bulletin 2000/41

(51) Int Cl.[7]: **G06F 9/46**

(21) Numéro de dépôt: **00400849.6**

(22) Date de dépôt: **28.03.2000**

(84) Etats contractants désignés:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Etats d'extension désignés:
AL LT LV MK RO SI

(30) Priorité: **07.04.1999 FR 9904337**

(71) Demandeur: **BULL S.A.**
78430 Louveciennes (FR)

(72) Inventeur: **Rogier, Pierre**
38410 Vaulnaveys le Haut (FR)

(54) Procédé d'amélioration des performances d'un Système multiprocesseur comprenant une file d'attente de travaux et architecture de système pour la mise en oeuvre du procédé

(57) L'invention concerne un procédé d'affectation de tâches dans un système de traitement de données numériques multiprocesseur à système d'exploitation préemptif et une architecture pour la mise en oeuvre de ce procédé. Le système comprenant des processeurs (200-203 et 210-213) susceptibles de traiter les tâches en parallèle répartis en groupes (200-201, 202-203). Une file d'attente élémentaire (5_a, 5_b) est associée à chacun des groupes de processeurs (200-201, 202-203) et enregistre des tâches à exécuter. Toutes les tâches à exécuter (T_1 à T_{10}) sont enregistrées dans une table (4). Chacune des tâches (T_1 à T_{10}) de la table (4) est associée à l'une des files d'attente (5_a, 5_b) et chacune des tâches enregistrées dans les files d'attente (5_a, 5_b) est associée à l'un des processeurs (200 à 201). Les associations sont effectuées par des jeux de pointeurs croisés (p_{200} à p_{203}, $pp5_a$, $pp5_b$, pT_1, pT_5, pT_{10}, p_{5a1} à p_{5a4}, p_{5b1} à p_{5b10}). Dans un mode de réalisation supplémentaire, selon plusieurs variantes, on procède à un (ré-)équilibrage de la charge du système entre files d'attente élémentaires.

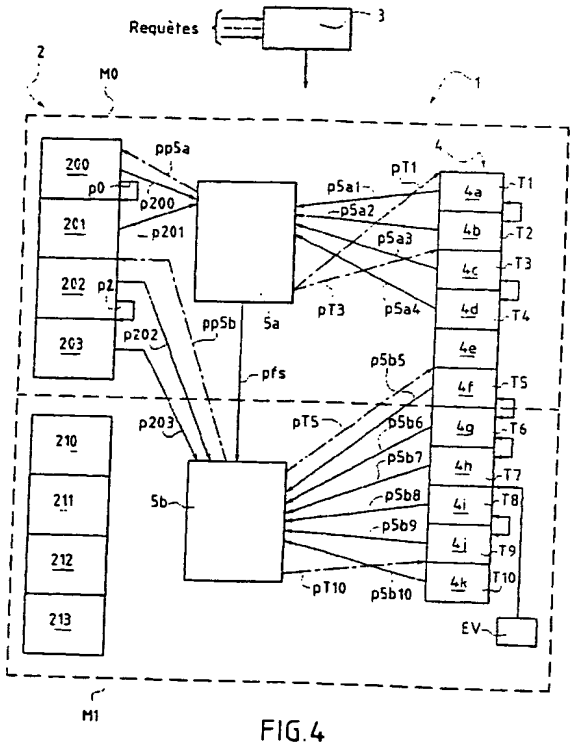


FIG.4

EP 1 043 658 A1

cesseurs à l'une des files d'attente élémentaires.

[0015] Cette disposition permet notamment de limiter le nombre de processeurs accédant aux verrous et donc de limiter le phénomène de contention.

[0016] Cependant, l'expérience montre que, lorsque le nombre de tâches et le nombre de processeurs augmentent, la disposition précitée ne permet plus d'améliorer les performances du système.

[0017] Ceci est dû à plusieurs phénomènes et notamment aux suivants :

[0018] Dans un système d'exploitation moderne existent deux types de tâches : les tâches à priorité variable et les tâches à priorité fixe. Les tâches du premier type sont des tâches dont la priorité varie en fonction du temps de processeur consommé (la politique d'ordonnancement est définie par le système d'exploitation lui-même). Les tâches du second type sont des tâches dont la politique d'ordonnancement est fixée lors de la définition de la tâche par le programmeur.

[0019] En premier lieu, la gestion des tâches de priorité fixe dans un système comportant plusieurs files d'attente, selon une première caractéristique du premier mode de l'invention, peut devenir complexe, car il est nécessaire d'éviter qu'une première tâche de priorité plus élevée soit exécutée après une seconde tâche de priorité moins élevée. Cette gestion s'avère en effet difficile, et surtout coûteuse en temps, lorsque les deux tâches précitées sont dans deux files d'attente distinctes. On conçoit aisément que cette difficulté augmente rapidement avec le nombre de tâches qui se répartissent dans un grand nombre de files d'attente.

[0020] Le problème subsiste pour des tâches de priorité variable, mais la difficulté de mise en oeuvre est moindre car c'est le système d'exploitation lui-même qui fixe les priorités, et il peut se permettre de violer ses propres règles.

[0021] En second lieu, le traitement des tâches peut devenir déséquilibré. Les tâches étant, *a priori,* de natures hétérogènes, le temps nécessaire au traitement de celles-ci peut varier dans de fortes proportions d'une tâche à l'autre. Il s'ensuit que un ou plusieurs processeurs, ou groupes de processeurs, peuvent se trouver en sous-charge, voire devenir inactifs, faute de tâches à traiter (les files d'attente associées s'étant vidées), alors qu'un ou plusieurs autres processeurs, ou groupes de processeurs, continuent de traiter des tâches (voire être surchargés) et qu'il reste des tâches à exécuter dans les files d'attente associées à ceux-ci. Aussi, dans un second mode de réalisation préféré de l'invention, tout en conservant les dispositions propres au premier mode de réalisation (partition des files d'attente), on procède à un rééquilibrage du traitement des tâches, selon plusieurs variantes.

[0022] Selon une première variante, le rééquilibrage comprend une répartition optimisée des tâches entre les différentes files d'attente du système. Le mode de répartition tient compte de différents paramètres qui seront précisés dans ce qui suit. La répartition peut être effectuée, soit lors de la création de la tâche, soit lors de l'association réalisée entre la tâche et un fichier contenant le programme à exécuter.

[0023] A titre d'exemple, dans un environnement de type "UNIX" précité, cette association est réalisée par une instruction de type : "*exec()*". Cette seconde option est préférable lorsque le système multiprocesseur est du type "NUMA' précité.

[0024] Cette disposition améliore les performances du système, y compris lorsque le nombre de tâches à traiter est très élevé. Cependant, la courbe représentant les performances présente des oscillations, qui traduisent des instabilités, notamment lorsque le nombre de tâches devient élevé. En outre, il est encore possible d'améliorer les performances.

[0025] Selon une deuxième variante de réalisation du second mode, lorsque la file d'attente associée à un processeur, ou à un groupe de processeurs, devient vide et que le processeur ou au moins l'un des processeurs n'a plus de tâche en cours de traitement, le processeur recherche dans les autres files d'attente s'il existe des tâches en attente de traitement. Si cette recherche est positive, dans un mode préféré, le processeur recherche ce qu'on pourra appeler la "meilleure tâche à traiter", s'il existe plusieurs tâches en attente. Le mode de recherche et de sélection de cette tâche sera précisé dans ce qui suit.

[0026] On doit bien comprendre que dans ces deux variantes, l'affectation des différentes tâches aux différentes files d'attente reste inchangée. L'association des première et deuxième variantes précitées est particulièrement efficace pour l'amélioration des performances du système tant que de nouvelles tâches se créent en permanence. Par contre, lorsque cet état cesse, par exemple en fin de travail du système, on peut être amené à constater de nouveau des déséquilibres de charge.

[0027] Aussi, l'invention peut comprendre une troisième variante de réalisation, dans laquelle on réaffecte des tâches entre différentes files d'attente, de façon périodique par exemple.

[0028] Cette disposition n'a généralement que peu d'effet en régime normal (création continue de tâches) sur les performances d'un système multiprocesseur symétrique, c'est-à-dire du type "SMP" précité. Elle peut cependant s'avérer utile pour un système de type "NUMA" précité.

[0029] L'invention a donc pour objet un procédé d'affectation de tâches dans un système de traitement de données numériques multiprocesseur, à système d'exploitation préemptif, comprenant un nombre déterminé de processeurs susceptibles de traiter lesdites tâches en parallèle, caractérisé en ce qu'il comprend au moins une phase préliminaire pendant laquelle lesdits processeurs sont répartis en groupes, chaque groupe comprenant des nombres prédéterminés

[0039] Un mécanisme dit "de verrou" est utilisé dans un certain nombre de circonstances, pour éviter un accès concurrent à une tâche, et notamment lorsqu'une tâche est ajoutée ou retirée de la file d'attente 5, ou lorsqu'elle change d'état.

[0040] On conçoit aisément que ce mécanisme de verrou global soit générateur de contentions lorsqu'il est fréquemment utilisé et n'autorise qu'une faible scalabilité. Cet inconvénient est amplifié lorsque le système multiprocesseur est du type "NUMA" précité.

[0041] Aussi, selon une caractéristique importante de l'invention, dans un premier mode de réalisation, on prévoit une partition de la file d'attente unique, et des verrous qui lui sont associés, en plusieurs ensembles de file d'attente et de verrous.

[0042] La figure 2 illustre schématiquement un exemple d'architecture de ce type. Le système 1 comprend, comme précédemment, plusieurs processeurs. Cependant, ces processeurs ont été rassemblés en groupes de processeurs, par exemple trois groupes référencés G_a à G_c. Chaque groupe, G_a à G_c, peut comprendre un nombre identique ou non de processeurs. Sur la figure 2, à titre d'exemple, on a supposé arbitrairement que le groupe G_a comprenait deux processeurs, 20_a et 21_a, le groupe G_b, trois processeurs, 20_b à 22_b, et le groupe G_c, un seul processeur, 20_c.

[0043] En outre, selon une première caractéristique importante de l'invention, la file d'attente unique (figure 1 : 5) est désormais divisée en une pluralité de files d'attente. De façon plus précise encore, le nombre de files d'attente est égal au nombre de groupes de processeurs, soit trois files d'attente dans l'exemple de la figure 2 : 5_a à 5_c, chaque file d'attente étant associée à l'un des groupes de processeurs, G_a à G_c.

[0044] En outre, et selon un autre aspect important, chaque tâche, T_1 à T_m, est affectée à une file d'attente particulière, 5_a à 5_c, et à une seule.

[0045] Ces affectations et associations s'effectuent, comme il le sera montré ci-après, en regard de la figure 4, à l'aide de jeux de pointeurs.

[0046] Le choix du nombre de groupes de processeurs, et donc du nombre de files élémentaires, dépend de nombreux paramètres dans un système multiprocesseur de caractéristiques données. Généralement, cette répartition ne peut pas être obtenue par des calculs préalables, mais par l'expérimentation et la mesure.

[0047] Le but que ce fixe l'invention est d'augmenter les performances globales du système par le biais d'une meilleure répartition des tâches entre processeurs individuels. Aussi, les expérimentations et tests précités consisteront, dans une phase initiale, à définir des programmes de test et de référence, dits "benchmark" selon la terminologie anglo-saxonne, et de les faire exécuter par le système. La répartition des processeurs en groupes associés à des files d'attente élémentaires donnant les meilleurs résultats, d'un point de vue performances, est retenue à ce stade. La configuration obtenue est généralement "gelée" et utilisée pour les systèmes de même structure fabriqués par la suite.

[0048] Il est d'ailleurs à présumer que, *a priori,* les meilleures performances devraient être atteintes en associant une file d'attente à chacun des processeurs. En d'autres termes, chaque groupe serait réduit à un seul processeur. Mais cette répartition peut engendrer des difficultés de réalisation. Aussi, un compromis est généralement préféré.

[0049] On va maintenant décrire de façon plus détaillée ce premier mode de réalisation du procédé de répartition des tâches selon l'invention.

[0050] Cependant, comme il a été indiqué, les architectures des systèmes multiprocesseurs du type "NUMA" accentuant les problèmes, on va se placer dans ce cadre et rappeler brièvement les caractéristiques principales d'une telle architecture par référence à la figure 3.

[0051] Le système 1 est divisé en modules, par exemple en deux modules, M_0 et M_1 comme représenté sur la figure 3 (ce nombre pouvant être quelconque). Chaque module, M_0 et M_1, comprend un nombre quelconque de processeurs pouvant fonctionner en parallèle. De façon pratique, le nombre de processeurs est limité à quelques unités, typiquement à quatre : 200 à 203 et 210 à 213, respectivement. En effet, lorsque le nombre de processeurs en parallèle augmente, les performances du système global augmentent tout d'abord sensiblement linéairement, puis la courbe s'infléchit. Le nombre quatre précité représente en général une valeur optimale. Les processeurs de chaque module, M_0 et M_1, sont connectés à des bus internes aux modules, B_0 et B_1 respectivement, et chaque module comprend notamment une mémoire centrale, Mem_0 et Mem_1. Les modules, M_0 et M_1, et leurs mémoires associées, Mem_0 et Mem_1, forment chacun un sous-système de type "SMP" précité. Les modules, M_0 et M_1, sont reliés entre eux par un lien L et un système de caches, C_1 et C_2, qui constituent un prolongement des bus internes précités.

[0052] On conçoit aisément que, par exemple, la lecture ou l'écriture d'une donnée de ou dans une mémoire externe à un module, par un processeur de ce module, se traduise par une dégradation des performances du système, par rapport à la même opération entièrement exécutée à l'intérieur d'un même module. Les performances sont également dégradées lorsque les données doivent transiter d'un module à l'autre par le lien qui ne peut généralement pas fonctionner à la même vitesse qu'un bus interne.

[0053] Aussi, on a proposé des procédés permettant d'obvier tout ou partie des problèmes spécifiques posés par les architectures de type "NUMA", procédés qui sortent du cadre précis de l'invention.

[0054] Cependant, le procédé de l'invention, dans son premier mode de réalisation, puisqu'il permet de limiter les contentions, du fait de la partition des files d'attente et des verrous associés, trouve une application particulièrement

chacune des files d'attentes, respectivement 5_a et 5_b : T_1-T_2 et T_3-T_4 pour la file d'attente 5_a, et T_5 à T_7 et T_{10}, pour la file d'attente 5_b. Les tâches T_8 et T_9 sont liées entre elles, mais sont à l'état "dormant". Il n'existe pas de pointeur associant ces tâches à la file 5_b dans cet état.

[0069] Comme il a été rappelé, il existe des tâches à priorité fixe et des tâches à priorité variable. Pour les tâches du premier type, il est obligatoire que l'ordre des priorités soit respecté, les tâches de plus fortes priorités devant être traitées avant les autres. Pour ce faire, on peut réserver une file d'attente aux tâches à priorité fixe. Toutefois, cette disposition n'est pas toujours envisageable. C'est le cas par exemple lorsqu'un processus comprend des tâches liées à un processeur donné. La tâche doit alors résider dans la file d'attente associée à ce processeur ou au groupe auquel il appartient. L'ordre des priorités est traité à l'intérieur de cette file d'attente.

[0070] En résumé de ce qui vient d'être décrit, le procédé selon l'invention, dans le premier mode de réalisation qui consiste à démultiplier les files d'attente, à affecter chaque file d'attente à un groupe de processeurs et à affecter chaque tâche à une file d'attente permet bien d'améliorer les performances globales du système. En effet les contentions sont réduites, car les verrous sont également distribués.

[0071] En outre, le procédé permet, dans une architecture de type "NUMA" d'implanter un mécanisme dit de "Weak Affinity", selon la terminologie anglo-saxonne. Un tel mécanisme favorise l'exécution d'une tâche sur un processeur d'un module unique, ce qui permet de tirer un plus grand bénéficie de la mémoire cache dite de "niveau 3" associée au module. Or, puisqu'il est possible d'associer une file d'attente à des processeurs appartenant à un seul module, le distributeur peut aisément confiner les tâches d'un processus dans un seul module.

[0072] Le procédé selon le premier mode de réalisation trouve cependant des limites lorsque le nombre de tâches et de groupes de processeurs augmentent fortement. En effet, lors de la création "physique" d'une tâche celle-ci doit être affectée à l'une des files d'attente du système, en faisant usage d'un mécanisme de distribution donné. Jusqu'à présent, il a été supposé implicitement que la distribution des tâches était réalisée sur une base d'équi-répartition temporelle entre les différentes files d'attente, dans la mesure où elles ne sont pas pleines. Pour ce faire, on peut utiliser un algorithme bien connu du type "round robin", c'est-à-dire à tour de rôle. Une telle méthode n'est pas sans inconvénients. En effet, dans les conditions précitées, les tâches présentant des caractéristiques non homogènes, notamment en ce qui concerne le temps de traitement nécessaire, il peut arriver qu'une ou plusieurs files d'attente soient vides ou peu chargées, et donc que les processeurs des groupes qui leur sont associés soient en sous-charge, voire inactifs pour le moins jusqu'à l'apparition de nouvelles tâches et à leur affectation à ces files d'attente. En sens inverse, une ou plusieurs autres files d'attente peuvent présenter des surcharges importantes. Il apparaît donc un phénomène de déséquilibre de charges, qui a d'autant plus de chance de se produire que le nombre de files d'attente et le nombre de tâches à traiter sont élevés. L'augmentation escomptée des performances globales du système est alors contrebalancée par ce phénomène parasite. Dans certains cas particulièrement défavorables, au-dessus d'un seuil donné de charge de travail, seuil qui dépend des ressources propres à un système particulier, il arrive de constater que les dispositions du procédé de l'invention sont contre-productives, en ce sens que les performances du système sont moins bonnes que celles d'un système de l'art connu présentant les mêmes ressources informatiques.

[0073] Aussi, selon un second mode de réalisation, mode de réalisation préféré, susceptible de plusieurs variantes, on adopte des dispositions supplémentaires permettant un (ré-)équilibrage de la charge entre les différentes files d'attentes, ou pour le moins un traitement optimisé des tâches réparties dans les files d'attente, de manière à ce que les processeurs soient utilisés de façon optimisée. On doit bien comprendre cependant que, selon ce second mode de réalisation, dans toutes ses variantes, les dispositions propres au premier mode sont conservées. Notamment, on répartit les processeurs en groupes (qui peuvent coïncider avec une répartition en modules dans un système d'architecture de type "NUMA") et on prévoit plusieurs files d'attente, une par groupe de processeurs.

[0074] Les opérations nécessaires pour obtenir cette configuration à files d'attente multiples, généralement effectuées une fois pour toutes, constituent donc une phase que l'on pourra qualifier de préliminaire. En mode opérationnel, un (ré-)équilibrage des tâches entre files d'attente ou de la charge de travail entre processeurs va être obtenu selon trois mécanismes, constituant précisément trois variantes du second mode de réalisation. Il doit d'ailleurs être noté que ces trois mécanismes peuvent coexister et ne sont pas exclusifs l'un de l'autre. Tout au contraire, dans un mode de réalisation préféré, on cumulera ces trois mécanismes, pour le moins les deux premiers qui donnent les meilleurs résultats quant aux objectifs poursuivis par l'invention, comme il le sera précisé ci-après.

[0075] Selon la première variante de réalisation, un équilibrage des tâches est obtenu en les répartissant de façon optimisée entre les différentes files d'attente lorsqu'elles apparaissent "physiquement", et non plus simplement à tour de rôle ("round robin" précité). La façon précise de choisir une file d'attente va être précisée ci-après.

[0076] Selon la deuxième variante de réalisation, un rééquilibrage du traitement des tâches est obtenu en optimisant l'utilisation effective des processeurs. Lorsqu'un processeur détecte que la file d'attente qui lui est associée est vide et qu'il n'a plus de tâche en cours de traitement, il va chercher une tâche à traiter dans une autre file d'attente, dite éloignée, en effectuant un balayage des autres files d'attente du système, jusqu'à ce qu'il trouve une file d'attente non vide et dont la charge est supérieure, *a priori*, à un seuil déterminé. Le choix d'une tâche précise dans la file d'attente sélectionnée, s'effectue selon un processus qui va être détaillé ci-après. De façon pratique, c'est le "dispatcher" qui

aussi calculée à chaque fois qu'une instruction d'exécution est initiée.

[0089] La variable *charge_FA#y* est une variable composite comprenant des constantes (*coef_charge_CPU#y* et *coef_charge_Mem#y*) qui peuvent être stockées dans une variable globale et sont susceptible d'être ajustée ("tunable") par l'administrateur du système pour obtenir un résultat optimum. Les autres composantes de la variable *charge_FA#y* sont déterminées elles-mêmes à partir de plusieurs paramètres décrivant le système, notamment à partir du nombre de tâches exécutables, de statistiques tenues à jour par le "scheduler" concernant les files d'attente et de l'occupation des mémoires, etc. Ces statistiques, pour la plupart, sont généralement disponibles dans les systèmes informatiques modernes et sont utilisées à d'autres fins que celles propres à l'invention. Le surcoût dû aux dispositions propres à l'invention, en terme de temps de calcul supplémentaire, est donc négligeable.

[0090] En ce qui concerne plus particulièrement les calculs permettant de déterminer la charge d'une mémoire, il peut être fait appel à des méthodes bien connues mettant en oeuvre des algorithmes d'estimation linéaire ou non linéaire

[0091] Lors de la création d'une nouvelle tâche T_z, et une fois que la file d'attente la moins chargée a été trouvée par l'organe 6, par exemple la file d'attente 5*y*, la nouvelle tâche T_z est aiguillée vers cette file d'attente par l'organe 6. Cet aiguillage a été symbolisé, sur la figure 5B, par un simple commutateur K.

[0092] Ces dispositions présentent de nombreux avantages, notamment les suivants :

a/ elles permettent de répondre très rapidement à des modifications également rapides du comportement du système 1 ;

b/ la détermination de la charge composite est simple, car basée sur deux valeurs qui peuvent être recherchées dans une même ligne de la mémoire cache de "niveau 2" ;

c/ le mécanisme n'est pas figé : il pourrait inclure d'autres variables, par exemple l'équilibrage de la charge de circuits d'entrées-sorties ("I/O")

d/ le déséquilibre dans des modules matériels est automatiquement pris en charge (c'est-à-dire le nombre de processeurs et/ou la taille mémoire) : en effet le nombre de processeurs est pris en compte du fait que le paramètre *moyenne_charge_Mem#y par_tâche* est relatif à une charge par processeur et la taille mémoire est prise en compte du fait que le nombre de pages mémoire (ou d'entités similaires) dépend de la taille mémoire ; et

e/ le mécanisme s'adapte de lui-même au jeu de ressources : s'il existe plusieurs files d'attente partageant un même pool de mémoire, la charge de la mémoire est la même pour tous les modules et seule la charge des processeurs est significative.

[0093] L'expérience montre que les dispositions propres à cette première variante du second mode permettent d'améliorer les performances globales du système, même en présence d'un grand nombre de files d'attente et de tâches à exécuter. Cependant, on peut constater dans certaines circonstances l'apparition d'instabilités. Par exemple, si l'on trace une courbe représentant le nombre de tâches exécutées par unité de temps (par exemple par heure) en fonction du nombre d'utilisateurs du système, ces instabilités se traduisent par des oscillations de la courbe.

[0094] On a représenté sur la figure 8 l'allure de la courbe représentant l'évolution du nombre de tâches exécutées par heure (chaque tâche étant représentée par exemple par un script) en fonction du nombre d'utilisateurs du système. La courbe *C* représente l'allure des performances d'un système non modifié, c'est-à-dire un système de l'art connu. La courbe C_A illustre le fonctionnement d'un système comportant les même ressources informatiques, mais dans lequel les dispositions propres à la première variante du second mode de réalisation du procédé de l'invention ont été implantées. On constate que la courbe C_A est (dans sa plus grande partie) située au-dessus de la courbe *C*, ce qui signifie que les performances ont été améliorées. Mais la courbe C_A présente des oscillations autour d'une position moyenne (représentée par une interpolation en traits interrompus C'_A). On constate aussi, dans l'exemple de la figure 8, que certaines oscillations font passer la courbe C_A en dessous de la courbe *C*. Pour ces portions de courbe, le système est moins performant qu'un système équivalent de l'art connu.

[0095] Aussi, il est préférentiellement fait appel à la deuxième variante de réalisation du second mode du procédé selon l'invention, dont les dispositions spécifiques peuvent se cumuler avec celles de la première variante.

[0096] Conformément à cette deuxième variante du second mode de réalisation du procédé selon l'invention, lorsqu'un processeur constate que la file d'attente qui lui est associée est vide et qu'il devient inactif, il va chercher une autre tâche exécutable dans une liste d'attente éloignée, non vide, ou pour le moins dont le taux de charge est supérieur à un seuil déterminé. Cependant, la tâche sélectionnée ne peut être quelconque. Elle doit répondre à certains critères qui vont être précisés ci-après.

[0097] La figure 6A illustre schématiquement un mode de recherche possible d'une tâche dans les files d'attentes du système 1. Les éléments communs aux figures précédentes portent les mêmes références et ne seront re-décrits qu'en tant que de besoin.

[0098] De façon habituelle, comme il a été montré en regard de la figure 4, les processeurs émettent des requêtes reçues par le "dispatcher" 3. On suppose ici que la file d'attente 5_q du processeur 2_q est vide et que celui-ci devient

[0110] En ce qui concerne le choix d'une tâche, un grand nombre de facteurs doivent être pris en compte, et parmi lesquels les suivants :

1/ l'affinité avec un processeur, c'est-à-dire le fait que la dernière distribution de la tâche s'est effectuée sur ce processeur ;
2/ l'affinité avec un module, dans le cas d'une architecture de type "NUMA", c'est-à-dire le fait que la dernière distribution de la tâche s'est effectuée sur ce module ;
3/ la priorité affectée à une tâche ;
4/ la localisation de la tâche ;
5/ le fait que la tâche ait déjà été "aidée" ;
6/ le fait que le processus soit mono-tâche ;
7/ la quantité de mémoire accédée par la tâche ;
8/ l'utilisation du processeur ; et
9/ la durée de vie de la tâche.

[0111] En ce qui concerne le facteur 3/ (priorité), il est préférable de "sauter" les tâche de plus haute priorité, c'est-à-dire les premières tâches dans la file d'attente "aidée". En effet, la probabilité est grande qu'elles soient prises en charge par un processeur local, du fait précisément de la haute priorité qui leur est associée, avant qu'elle puisse être traitée par le processeur éloigné. L'utilisation d'un seuil prédéterminé semble être une solution appropriée pour cette partie du processus. En outre, les tâches de plus faibles priorité sont généralement, en moyenne statistique, des tâches qui utilisent le plus le processeur.

[0112] La détermination d'une valeur de seuil est importante. En effet, si la valeur de seuil est trop basse, c'est-à-dire que le nombre de tâches sautées est trop faible, le mécanisme d'aide se trouve alors souvent en conflit avec le mécanisme standard de distribution des tâches, c'est-à-dire le mécanisme commun à l'art connu. En sens contraire, si le seuil est fixé à une valeur trop haute, aucune tâche ne pourra être trouvée et le mécanisme d'aide s'avère complètement inefficace.

[0113] De façon préférentielle, pour être aussi indépendant que faire ce peut de la charge de travail, on met en oeuvre un procédé auto-adaptatif, par exemple le suivant :

[0114] Le nombre de tâches sautées est fixé à une valeur comprise entre le nombre de processeurs et le nombre de tâches exécutables dans l'ensemble de file d'attente. Cette valeur est incrémentée d'une unité chaque fois que la tâche choisie pour être "aidée" est, soit déjà verrouillée, soit n'est pas à l'état exécutable. Cette valeur est décrémentée d'une unité chaque fois qu'aucune tâche n'est trouvée, lorsque le nombre maximum de tâches à balayer est supérieur à la moitié du nombre de tâches exécutables.

[0115] Le nombre maximum de tâches à balayer est fixé à une valeur comprise entre l'unité et le nombre de tâches exécutables dans l'ensemble de file d'attente. Cette valeur est incrémentée d'une unité chaque fois qu'aucune tâche n'est trouvée ou chaque fois que la tâche choisie se trouve dans le dernier quart des tâches balayées (tâches de plus basses priorités). Cette valeur est décrémentée d'une unité chaque fois que la tâche choisie se trouve dans le premier quart des tâches balayées (tâches de plus hautes priorités).

[0116] Le facteur 4/ (localisation) est, *a priori*, un facteur très important. Cependant, ce facteur est généralement difficile à déterminer, bien que, dans un environnement de type "UNIX", la localisation de la tâche soit connue par segment de mémoire.

[0117] En ce qui concerne le facteur 5/, on peut généralement admettre que, si une tâche a déjà été "aidée", elle peut déjà résider dans plusieurs modules. Il s'ensuit que de la déplacer ailleurs ne constitue pas une opération coûteuse en terme de dégradation de performances.

[0118] En ce qui concerne le facteur 7/, il s'agit également d'un facteur important, mais qui ne peut être déterminé aisément. Deux critères permettent d'arriver à une approximation raisonnable :

a/ la taille mémoire utilisée par le processus ; et
b/ "l'interactivité" de la tâche, ce critère étant défini par le fait qu'une tâche soit souvent "dormante" ou non.

[0119] Le critère b/ peut être obtenu à partir d'un comptage du nombre de fois où elle est à l'état "dormant", ce qui peut se dériver de statistiques généralement disponibles.

[0120] En ce qui concerne enfin le facteur 9/, il est aisé de comprendre qu'il n'est pas utile de tenter de prendre en charge les tâches de faible durée de vie. En effet, la plupart d'entre elles disparaissent à court terme.

[0121] En tenant compte de tout ou partie de ces différents facteurs, il est possible de déterminer quelle tâche doit être sélectionnée dans une file d'attente, en définissant un coût individuel associé à chaque facteur, et d'en déduire un coût global associé à une tâche particulière. On peut, pour ce faire construire une table à deux entrées : facteurs - coûts. La tâche présentant le coût global le plus faible est sélectionnée, c'est-à-dire celle causant la dégradation

- si $\|\overrightarrow{CLP}-\overrightarrow{AD}\| < \|\overrightarrow{AD}\|$ trouver la file d'attente d'indice arbitraire y pour laquelle le paramètre $\|\overrightarrow{CLP} - \overrightarrow{AD}\|$ est minimal ;
- migrer ce processus vers l'ensemble de file d'attente le moins chargé 5_y ; et
- mettre à jour le facteur représentatif du déséquilibre des deux ensembles de file d'attente, soit $\overrightarrow{AD}_x = \overrightarrow{AD}_x - \overrightarrow{CLP}_x$ et $\overrightarrow{AD}_y = \overrightarrow{AD}_y - \overrightarrow{CLP}_y$.

[0134] Le vecteur de charge composite est un vecteur à trois dimensions. En effet, il dépend des paramètres suivants :

- charge du processeur ;
- charge de la mémoire ; et
- priorité.

[0135] Les deux premiers paramètres dépendent à leur tour de la configuration matérielle et logicielle précise du système considéré : nombre de processeurs, taille de la mémoire, nombre de pages mémoire libres, etc. La détermination de ces paramètres est commune à l'art connu et obtenue par des calculs classiques, biens connus de l'homme de métier. Le paramètre "priorité" est obtenu à partir de la moyenne des priorités attachées aux différentes tâches.

[0136] Théoriquement, la détermination de la charge d'un ensemble de file d'attente est donnée par la somme des charges de processus. Mais pour accélérer cette détermination, on la dérive directement à partir de statistiques généralement stockées dans la structure de données de cet ensemble. La charge dépend de nouveau de trois paramètres : charge du processeur, charge de la mémoire et priorité.

[0137] La détermination de la charge composite moyenne peut être obtenue à partir de la relation suivante :

$$\overrightarrow{ACL} = \frac{\sum_{i=1 \text{ à } p} \overrightarrow{CL_i}}{p} \qquad (3)$$

relation dans laquelle est la charge composite du $i^{ème}$ ensemble de file d'attente et p le nombre total d'ensembles de file d'attente.

[0138] Le déséquilibre moyen peut être déterminé à partir de la relation suivante :

$$\overrightarrow{AD_i} = \frac{\overrightarrow{AD_i} + \left(\overrightarrow{CL_i} - \overrightarrow{ACL_i}\right)}{2} \qquad (4)$$

[0139] La détermination du coût associé à une opération de migration peut être obtenue en considérant que le coût principal est dû à la migration de pages de mémoires, dans un environnement de type "UNIX" (ou à l'accès à des pages éloignées) et au coût lié au déplacement d'une tâche d'un ensemble de file d'attente à un autre.

[0140] Une approximation de l'estimation du coût est obtenue directement par le nombre de pages associées au processus et par le nombre de tâches devant être déplacées. Dans un environnement autre que l'environnement "UNIX", l'entité "page de mémoire" doit être remplacé par une entité équivalente.

[0141] Ces modes de détermination des paramètres impliqués ne sont précisés qu'à titre d'exemple, pour fixer les idées. D'autres alternatives existent et sont à la portée de l'homme de métier.

[0142] Si on se reporte de nouveau à la figure 8, la courbe C_C illustre schématiquement l'allure de l'amélioration des performances par rapport à l'art connu (courbe C). Cependant, l'expérience montre que, dans le cas général, l'amélioration obtenue par rapport à celle obtenue par la deuxième variante est peu importante. Ceci est dû essentiellement au fait que le déplacement physique des tâches entre files d'attente implique un coût non négligeable, ce même si celui-ci n'est pas généralisé, conformément aux dispositions préférentielles qui viennent d'être rappelées, mais au contraire sélectif. On réservera cette variante du procédé selon l'invention à une architecture de type "NUMA", car dans le cas d'une architecture classique de type "SMP", l'amélioration des performances n'est pas significative, alors que sa mise en oeuvre nécessite des modifications du système d'exploitation et la présence d'organes supplémentaires, matériel ou logique (figure 7 : 8).

[0143] A la lecture de ce qui précède, on constate aisément que l'invention atteint bien les buts qu'elle s'est fixés.

[0144] Il doit être clair cependant que l'invention n'est pas limitée aux seuls exemples de réalisations explicitement décrits, notamment en relation avec les figures 2 et 4 à 8.

de charge composite est calculé comme étant la somme de la charge d'un processeur ou d'un groupe de processeurs associé à ladite file d'attente élémentaire et la charge des moyens de mémoire associés à ce processeur ou ce groupe de processeurs.

8. Procédé selon la revendication 6, caractérisé en ce qu'il comprend une étape préalable consistant à tester si ladite nouvelle tâche (Tz) est liée à une desdites files d'attente élémentaires (5_a, 5_x, 5_y, 5_p) et en ce que lorsque ledit test est positif à aiguiller ladite nouvelle tâche vers cette file d'attente élémentaire.

9. Procédé selon l'une quelconque des revendications 1 à 5, caractérisé en ce qu'il comprend au moins une phase supplémentaire consistant, lorsque l'une desdites files d'attente élémentaires (5_q) associée à l'un desdits groupes de processeurs (2_q) est vide de tâches exécutables, à rechercher une file d'attente élémentaire (5_y), dite éloignée, non vide et dans cette file d'attente élémentaire (5_y) à sélectionner une tâche exécutable par l'un desdits processeurs (2_q) dudit groupe de processeurs associé à la file d'attente élémentaire vide (5_q) et à la transmettre à ce processeur (2_q) pour y être traitée, de manière à équilibre globalement le traitement desdites tâches dans ledit système (1).

10. Procédé selon la revendication 9, caractérisé en ce que ladite file d'attente élémentaire non vide (5_y) doit présenter un seuil d'occupation minimal prédéterminé.

11. Procédé selon la revendication 10, caractérisé en ce qu'en outre, les tâches étant enregistrées par ordre de priorité décroissante, un nombre prédéterminé de tâches est sauté avant de balayer les autres tâches de ladite file d'attente élémentaire non vide (5_y) pour rechercher une tâche exécutable et la faire traiter par l'un desdits processeurs (2_q) dudit groupe de processeurs associé à la file d'attente élémentaire vide (5_q).

12. Procédé selon la revendication 11, caractérisé en ce que ledit nombre de tâches sautées et le nombre maximum de tâches balayées parmi toutes celles enregistrées dans ladite file d'attente élémentaire non vide (5_q) sont variables dans le temps et sont déterminés par un processus auto-adaptatif à partir du nombre de tâches trouvées ou non pendant lesdits balayages et de la place de ces tâches classées par ordre de priorité dans ladite file élémentaire non vide (5_q).

13. Procédé selon l'une quelconque des revendications 9 à 12, caractérisé en ce que ladite tâche sélectionnée est celle associée à une valeur minimale d'un paramètre dit de coût, mesurant la dégradation de performances globale dudit système (1), due au traitement de ladite tâche sélectionnée dans ladite file d'attente élémentaire éloignée non vide (5_q) par l'un desdits processeurs dudit groupe de processeurs associé à la file d'attente élémentaire vide (2_q).

14. Procédé selon l'une quelconque des revendications 1 à 5, caractérisé en ce qu'il comprend au moins une phase supplémentaire comprenant au moins une étape de mesure périodique d'une répartition équilibrée desdites tâches dans lesdites files d'attente élémentaires (5_a, 5_x, 5_y, 5_p) et, lors de la détermination d'un état déséquilibré dudit système (1), une étape de déplacement sélectif de tâches d'au moins une file d'attente élémentaire plus chargée (5_x) vers une file d'attente élémentaire moins chargée (5_y).

15. Procédé selon la revendication 14, caractérisé en ce que, lorsque ledit déséquilibre est inférieur à un seuil déterminé, aucun déplacement de tâche n'est réalisé.

16. Procédé selon les revendications 14 ou 15, caractérisé en ce que tout ou partie desdites tâches appartenant à des processus multitâches, chaque processus multitâche nécessitant une taille de mémoire et une charge de travail déterminées, il comprend une étape de mesure desdites charges de travail et desdites tailles de mémoire, en ce qu'il comprend la sélection du processus nécessitant la plus forte charge de travail et la plus faible taille de mémoire, et en ce que ce que toutes les tâches dudit processus sélectionné sont déplacées vers la file d'attente élémentaire la moins chargée (5_y).

17. Procédé selon la revendication 16, caractérisé en ce qu'il comprend une étape préliminaire consistant à tester si toutes les tâches dudit processus multitâche devant être déplacées appartiennent à l'ensemble de file d'attente élémentaire le plus chargé (5_x) et si aucune tâche n'est liée à l'un desdits groupes.

18. Procédé selon l'une quelconque des revendications 1 à 17, caractérisé en ce que ledit système d'exploitation est du type "UNIX" (marque déposée).

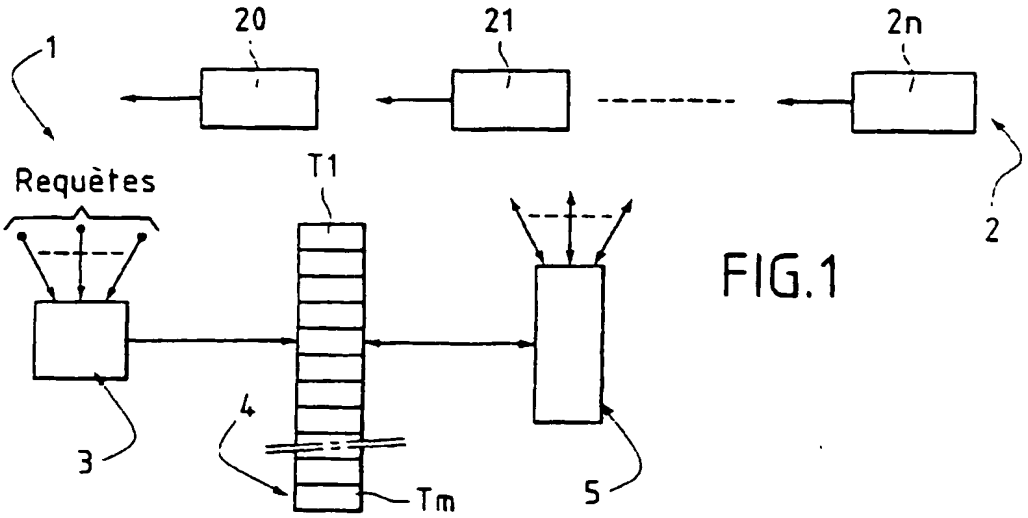


FIG.1

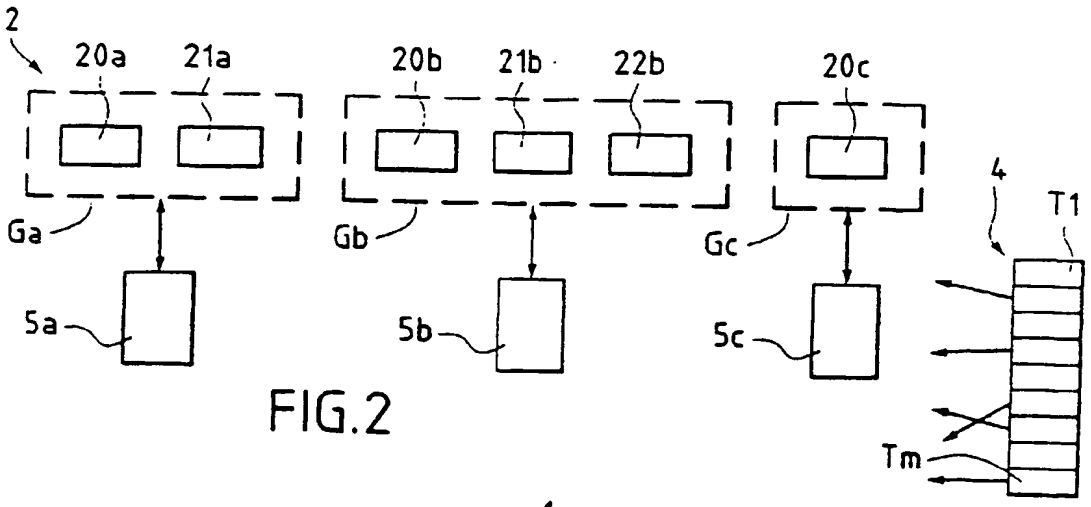


FIG.2

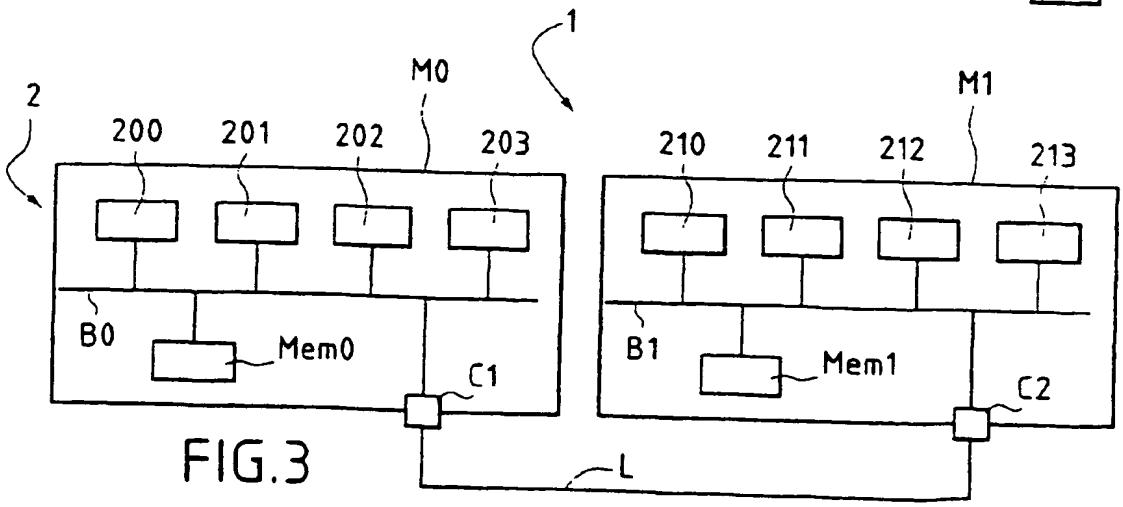


FIG.3

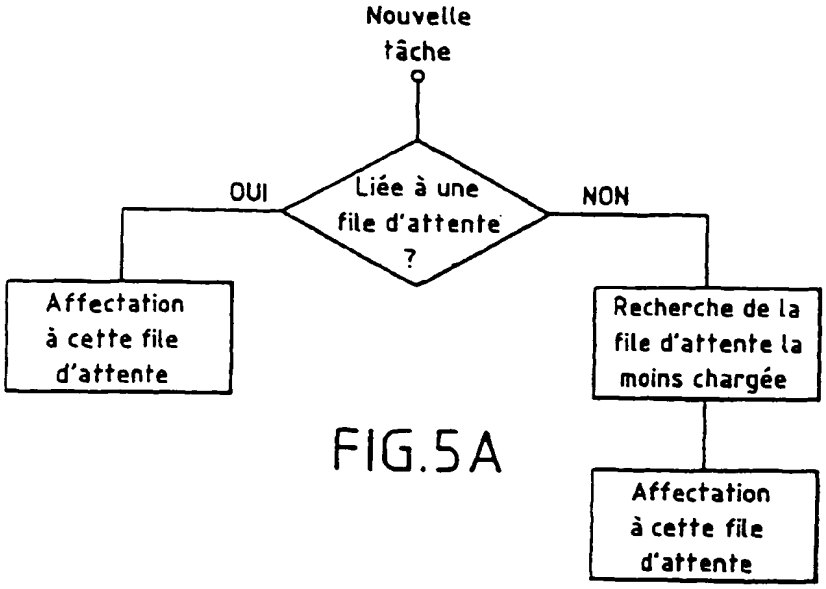


FIG.5A

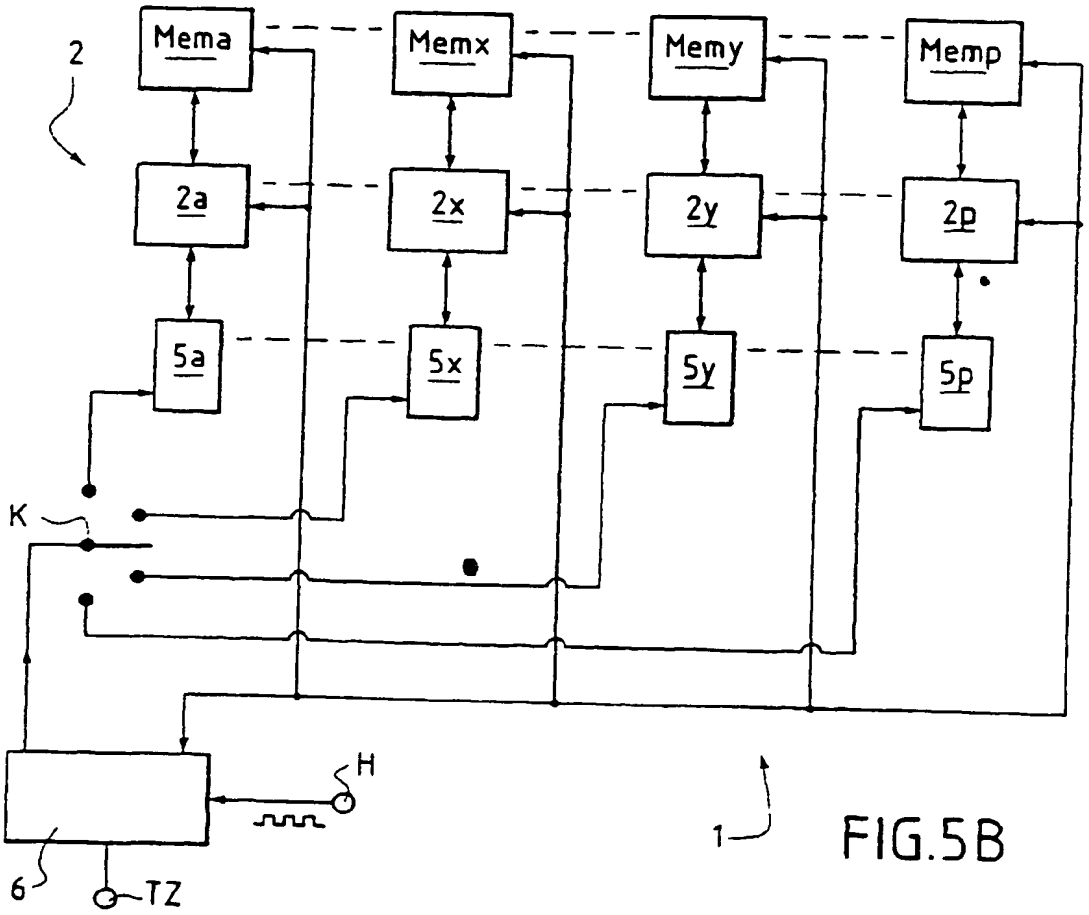


FIG.5B

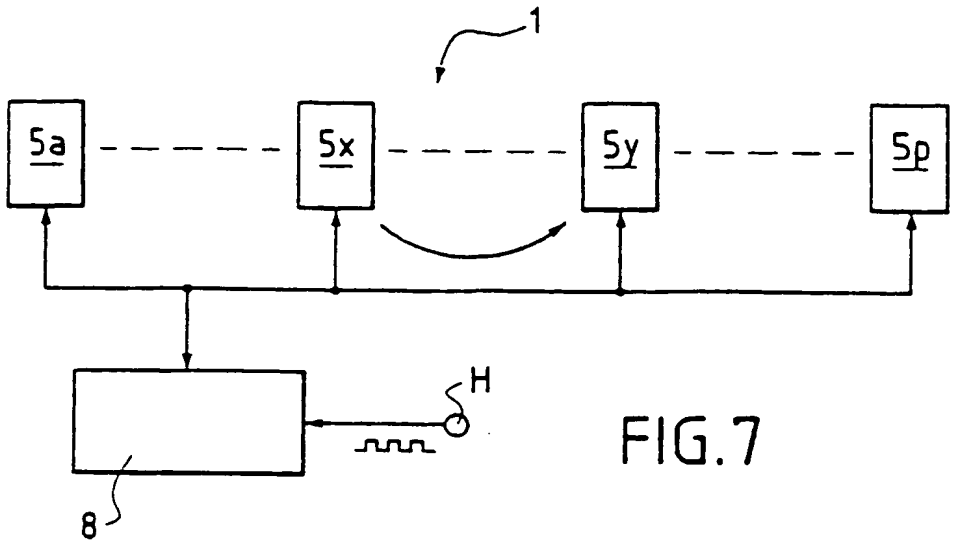
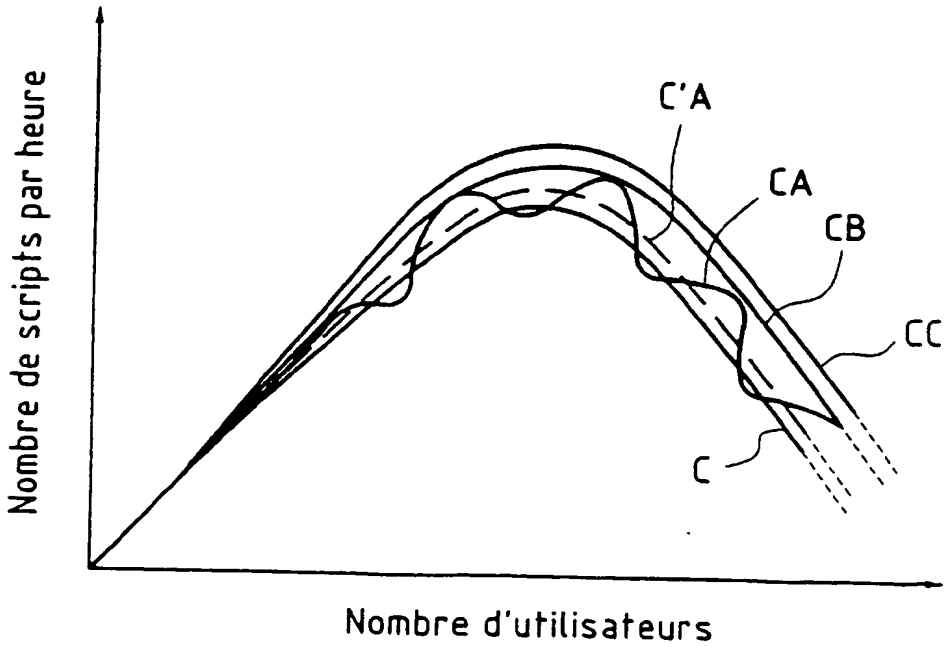


FIG.7

FIG.8



ANNEXE AU RAPPORT DE RECHERCHE EUROPEENNE RELATIF A LA DEMANDE DE BREVET EUROPEEN NO. EP 00 40 0849

La présente annexe indique les membres de la famille de brevets relatifs aux documents brevets cités dans le rapport de recherche européenne visé ci-dessus.
Lesdits membres sont contenus au fichier informatique de l'Office européen des brevets à la date du
Les renseignements fournis sont donnés à titre indicatif et n'engagent pas la responsabilité de l'Office européen des brevets.

28-07-2000

Document brevet cité au rapport de recherche	Date de publication	Membre(s) de la famille de brevet(s)	Date de publication
FR 2740579 A	30-04-1997	JP 2940450 B	25-08-1999
		JP 9120389 A	06-05-1997
		US 5881284 A	09-03-1999
EP 0750255 A	27-12-1996	AU 701540 B	28-01-1999
		AU 5601496 A	09-01-1997
		CA 2179483 A	24-12-1996
		JP 9237215 A	09-09-1997
US 5459864 A	17-10-1995	JP 2587195 B	05-03-1997
		JP 6250983 A	09-09-1994

EPO FORM P0460

Pour tout renseignement concernant cette annexe : voir Journal Officiel de l'Office européen des brevets, No.12/82